LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS

Hu 等 – 2021 – LoRA Low-Rank Adaptation of Large Language Models.pdf

github repo:

> GitHub – microsoft/LoRA: Code for loralib, an implementation of "LoRA: Low-Rank Adaptation of Large Language Models"

摘要

自然语言处理的一个重要范式是在通用领域数据上进行大规模预训练，然后适应特定任务或领域。随着预训练模型规模的增大，全参数微调（即重新训练所有模型参数）变得越来越不可行。以GPT-3 175B为例——部署每个具有1750亿参数的独立微调模型实例，成本高得令人望而却步。我们提出低秩适应（Low-Rank Adaptation，简称LoRA），该方法冻结预训练模型权重，并在Transformer架构的每一层中注入可训练的低秩分解矩阵，从而大幅减少下游任务的可训练参数数量。与使用Adam进行微调的GPT-3 175B相比，LoRA可将可训练参数减少一万个倍，并将GPU内存需求降低三倍。尽管可训练参数更少、训练吞吐量更高，且与适配器不同，不会引入额外的推理延迟，LoRA在RoBERTa、DeBERTa、GPT-2和GPT-3上的模型质量表现与微调相当甚至更优。我们还对语言模型适应中的低秩缺陷进行了实证研究，揭示了LoRA的有效性。我们发布了一个便于将LoRA集成到PyTorch模型的工具包，并在https://github.com/microsoft/LoRA上提供了RoBERTa、DeBERTa和GPT-2的实现与模型检查点。

问题背景

自然语言处理的一个重要范式是在通用领域数据上进行大规模预训练，然后适应特定任务或领域。随着预训练模型规模的增大，全参数微调（即重新训练所有模型参数）变得越来越不可行。微调参数多，成本高。

> ❌许多人试图通过仅调整部分参数或为新任务学习外部模块来缓解这一问题。现有的技术通过增加模型深度引入推理延迟，或减少模型可用的序列长度，这些方法通常难以达到微调基线的性能

解决方法

✅低秩适配（LoRA）方法：LoRA允许我们通过优化适配过程中稠密层变化的低秩分解矩阵来间接训练神经网络中的部分稠密层，同时保持预训练权重冻结（不会改变模型的结构，只会微调参数）。

LORA优点：

- **预训练模型可以被共享，并用于为不同任务构建多个小型LoRA模块。**
- **LoRA通过使用自适应优化器使训练更加高效**
- **不会增加推理延迟**

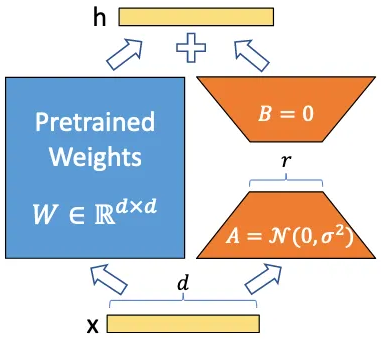


Figure 1: Our reparametrization. We only train A and B.

问题陈述

lora对训练目标是中立的，对于全参数进行微调是不现实的，我们引入了lora

$$\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t \mid x, y_{<t})\right)$$

(x,y) 表示训练样本，其中 x 是输入，y 是输出序列。 $\sum_{t=1}^{|y|}$ 对输出序列 y的每一个 token 进行求和。

$\log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t \mid x, y_{<t})\right)$ 表示模型参数 $\Phi_0 + \Delta\Phi(\Theta)$ 下，预测第 t 个 token 的条件概率。

总结：找到最优的 Θ ，使得模型在训练数据上生成目标序列的概率最大。

现有方法的缺点

在高效适配方面有两种突出的策略：添加适配器层、或优化输入层激活的某种形式，但是都有其局限性。

1. **适配器层引入推理延迟：**大型神经网络依赖硬件并行性以维持低延迟，而适配器层必须**串行处理**
2. **直接优化提示词是困难的：**前缀微调难以优化，且其性能随可训练参数的变化呈现**非单调性**

拓展+参数高效微调PEFT

- Adapter tuning：是在基础模型的各层之间插入小型神经模块，训练时只更新这些 Adapter 参数，原始大模型参数保持冻结。 $h' = h + W_{\text{up}} f(W_{\text{down}} h)$ ， W_{down} ：把高维隐藏向量降到低维，f：非线形激活函数， W_{up} 再升到高维，并使用残差连接保证稳定性
- Prefix tuning：但提示不只是加在输入 embedding 上，在每一层的 Self-Attention 里，额外引入一段“前缀键值对” (K_{prefix}, V_{prefix})

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q\left[K; K_{prefix}\right]^{\top}}{\sqrt{d}}\right)\left[V; V_{prefix}\right]$$

- Prompt Tuning: 修改模型的输入，在模型输入的前面加一些特定的前缀

改进的方法

预训练语言模型具有较低的“内在维度”，即使通过随机投影到更小的子空间，仍能高效学习。引入了LORA。

$$h = W_0 x + \Delta W x = W_0 x + \frac{\alpha}{r} B A x, \quad B \in \mathbb{R}^{d \times r}, \ A \in \mathbb{R}^{r \times k}$$

α 通常是一个超参数（类似学习率），越大权重越高，适配能力更强，但可能不稳定，小则收敛慢

r = 秩，控制 LoRA 矩阵的容量（结构超参数）。跟上面维度r是一个东西

初始化：A使用随机高斯初始化（用一个指定均值和方差的正态分布，随机生成权重矩阵的初始值），对B使用零初始化

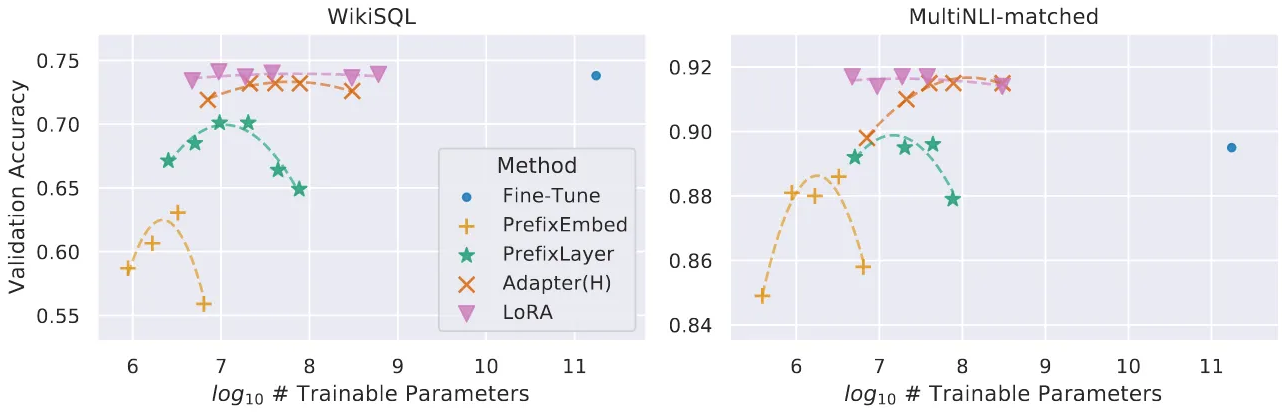
当我们需要切换到另一个下游任务时，可以通过减去 BA 再加上不同的 B′A′ 来恢复 W0

实验验证

FT、BitFit、PreEmbed、PreLayer、Adapter H等baseline和**LORA**进行对比性实验

Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 ($\text{Adapter}^{\text{H}}$)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 ($\text{Adapter}^{\text{H}}$)	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

超大参数1750亿参数的GPT-3：LoRA 展现出更好的可扩展性和任务性能。



提出问题

1. **在给定参数预算约束下，我们应调整预训练Transformer中哪一组权重矩阵以最大化下游任务性能?**

	# of Trainable Parameters = 18M						
Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL (±0.5%)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI (±0.1%)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

调整 W_v 和 W_q 的整体表现最佳。

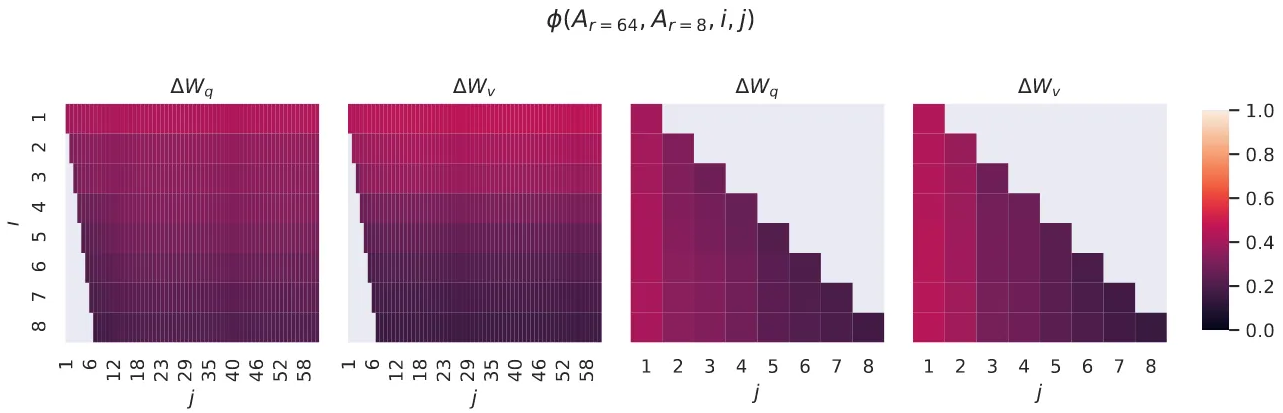
2. **“最优”适应矩阵ΔW是否确实为低秩？如果是，实践中采用何种秩较为合适?**

	Weight Type	$r=1$	$r=2$	$r=4$	$r=8$	$r=64$
WikiSQL($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

不同 r 下的子空间相似性：

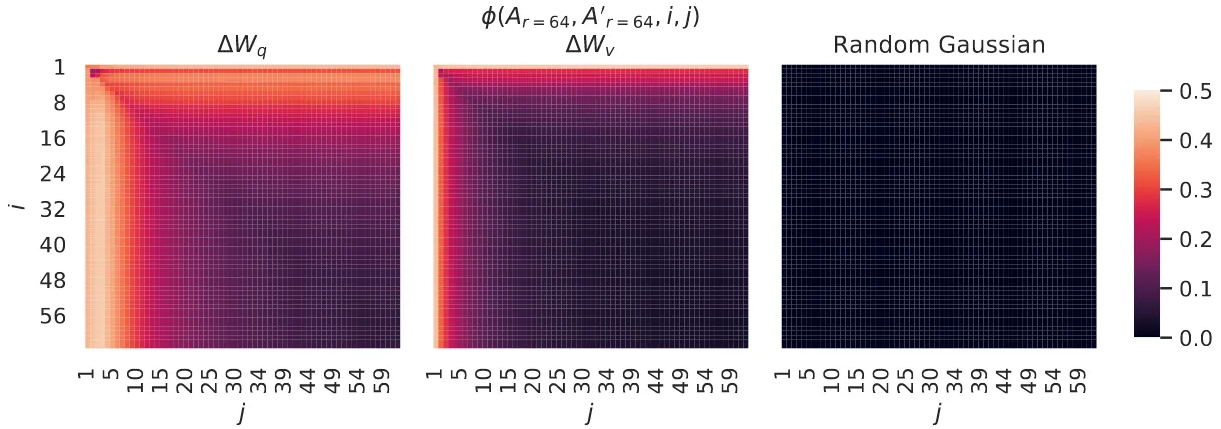
$$\phi(A_{r=8}, A_{r=64}, i, j) = \frac{\|U^{i\top}_{A_{r=8}} U^{j}_{A_{r=64}}\|_F^2}{\min(i,j)} \in [0,1]$$

$U^{i\top}_{A_{r=8}} U^{j}_{A_{r=64}}$ ：对矩阵 $A_{r=8}, A_{r=64}$ 做**矩阵分解**（如 SVD 奇异值分解）后得到的“左奇异矩阵的子矩阵”。 $\min(i,j)$ 进行归一化处理，**衡量两个低秩矩阵“相关性”或“近似程度”的指标**



反映了小主成分数量时相关性强，大主成分数量时相关性弱。

3. ΔW与W之间存在何种关联？ΔW是否与W高度相关？ΔW相较于W的规模有多大？



模型权重变化相关的矩阵 ΔW_q 、 ΔW_v 具有结构化的子成分相关性，而随机高斯矩阵无此特性”，反映了模型权重更新的“非随机、结构化”本质.